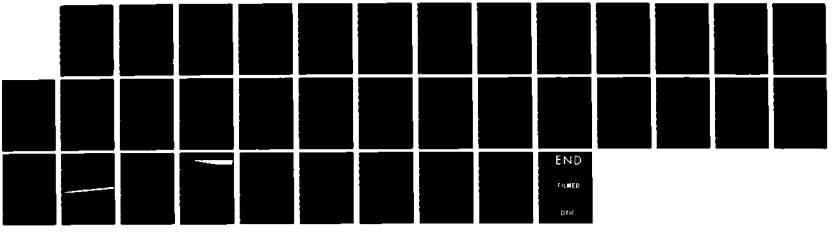
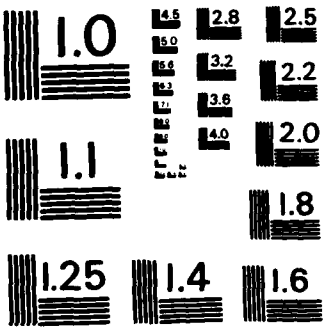
AD-A160 314 ADAPTIVE REFINEMENT METHODS FOR NONLINEAR PARABOLIC PARTIAL DIFFERENTIAL.. (U) RENSSELAER POLYTECHNIC INST TROY NY DEPT OF MATHEMATICAL SCIE.. 1/1

UNCLASSIFIED M BIETERMAN ET AL. DEC 84 AFOSR-TR-85-0827 F/G 12/1 NL





Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

AD-A160 314

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S): N/A

5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 85-0827

6a. NAME OF PERFORMING ORGANIZATION: Rensselaer Polytechnic Institute

6b. OFFICE SYMBOL (If applicable):

7a. NAME OF MONITORING ORGANIZATION: AFOSR

6c. ADDRESS (City, State and ZIP Code): Department of Mathematical Sciences, Troy, NY 12181

7b. ADDRESS (City, State and ZIP Code): Bolling AFB, D.C. 20332-6448

8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR

8b. OFFICE SYMBOL (If applicable): NM

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-80-0192

8c. ADDRESS (City, State and ZIP Code): Bldg. 410, Bolling AFB, D.C. 20332-6448

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	2304	A3	

11. TITLE (Include Security Classification): Adaptive refinement methods for nonlinear parabolic partial differential equations

12. PERSONAL AUTHOR(S): M. Bieterman, J. E. Flaherty, and P. K. Moore

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Interim	FROM _____ TO _____	December 1984	34

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
	XXXXXXXXXX	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Adaptive methods, finite element methods, method of lines, local refinement methods, parabolic differential equations

19. ABSTRACT (Continue on reverse if necessary and identify by block number):

In this chapter we considers two adaptive finite element techniques for parabolic partial differential equations (PDEs) that are based on using error estimates to control mesh refinement. One technique is a method of lines (MOL) approach that uses a Galerkin method to discretize the PDEs in space and implicit multi-step integration in time. Spatial elements are added and deleted in regions of high and low error and are all advanced with the same sequence of varying time steps. The The second technique is a local refinement method (LRM) that uses Galerkin approximations in both space and time. Fine grids of space-time elements are added to coarser grids and the problem is recursively solved in regions of high error.

DTIC FILE COPY

DTIC ELECTE OCT 15 1985 S A

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☒ DTIC USERS ☐

21. ABSTRACT SECURITY CLASSIFICATION: Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Brian W. Woodruff MAJ, USAF	(202)767-5027	NM

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

ADAPTIVE REFINEMENT METHODS FOR NONLINEAR PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS

MICHAEL BIETERMAN
Boeing Computer Services
Engineering Technology Applications Division
Tukwila, WA 98188 USA

JOSEPH E. FLAHERTY
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York 12181 USA

PETER K. MOORE
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, New York 12181 USA

1. INTRODUCTION

In this chapter we consider two adaptive finite element techniques for parabolic partial differential equations (PDEs) that are based on using error estimates to control mesh refinement. One technique is a method of lines (MOL) approach that uses a Galerkin method to discretize the PDEs in space and implicit multi-step integration in time. Spatial elements are added and deleted in regions of high and low error and are all advanced with the same sequence of varying time steps. The second technique is a local refinement method (LRM) that uses Galerkin approximations in both space and time. Fine grids of space-time elements are added to coarser grids and the problem is recursively solved in regions of high error.

In describing these methods we shall not concentrate on algorithmic details or provide extensive numerical results. The MOL has evolved over the past few years, and readers interested in examining related theory and more extensive experiments may wish to see Refs. 7, 8, and 9. The LRM has been studied for a much shorter period of time, but readers interested in directly

related methods may wish to see Refs. 4, 5, 6, and 12. There is an ever expanding number of adaptive methods for time-dependent PDEs appearing in the literature. We refer a reader unfamiliar with this area especially to the other chapters of this text and to Ref. 2.

Our intention is to highlight some of the difficulties faced by many adaptive methods for time-dependent PDEs by comparing and contrasting the MOL and LRM in a simple manner. These methods differ in many respects, but they also have some similiar features.

Both are applied to m-dimensional vector systems of PDEs of the form

$$Lu := u_t + f(x,t,u,u_x) - [D(x,t,u)u_x]_x = 0 ,$$

$$a < x < b,\ 0 < t \le T \qquad (19.1)$$

with appropriate initial and separated boundary conditions. Such systems model many physical phenomena, including heat conduction, flame propagation in combustion, signal transmission in nerves, and contaminant transport in porous media.

In approximating the solution of Eq. (19.1), an attempt is made in both methods to efficiently measure and control the accuracy of the computations. The methods' basic objectives are:

i. Obtain an accurate estimate E(t) of $||e(t)||$ for all $t \in (0,T)$, where e is the total computational error and $||\cdot||$ is a spatial norm for the MOL and a spatial norm with local temporal variations for the LRM.

ii. For a given tolerance TOL > 0, adjust the space and time discretizations so that $||e(t)|| \le$ TOL for all $t \in (0,T)$.

iii. Keep total computational cost and storage overhead low.

The MOL and LRM adopt the same general approach for resolving the conflict that exists between the third and the first two objectives. Error estimates are formed from indicators, which are derived from local values of the computed solution and are discarded as integration progresses on (a,b) x (0,T). The error control strategies are based on direct control of the error estimates and on relatively little additional information. One could thus expect this approach to be less successful in achieving the first two objectives than a method which integrates auxiliary equations for the error or which uses data stored in previous integrations, but more successful in achieving the third. Alternatively, an approach not employing error estimates and an error tolerance refinement criterion like those here might be more successful in maximizing accuracy per cost or storage, but it would have no direct control over the level of accuracy.

Both of the methods considered employ "coarse" space and time grids

$$\Delta_x := \{a = X_0 < X_1 < \ldots < X_N = b\} \tag{19.2}$$

and

$$\Delta_t := \{0 = t_0 < t_1 < \ldots < t_K = T\} \tag{19.3}$$

to guide the mesh refinement process. These grids are uniform and are chosen a priori to reflect expected "global" scales on which the solution of Eq. (19.1) will vary. Integration in the MOL and LRM consists of sequential application of a core algorithm for Eq. (19.1) on the space-time strips $\{(a,b)\times(t_k,t_{k+1})\}_{k=0,\ldots,K-1}$. The core algorithms differ significantly in

form.

The MOL algorithm for $(a,b)\times(t_k,t_{k+1})$ begins at t_k with a space grid δ containing Δ_x and with solution values at the nodes of δ. Spatial discretization of Eq. (19.1) is completed with a standard piecewise-linear finite element formulation, which results in an ordinary differential equation initial value problem (ODE-IVP) for solution values on lines extending in time from the nodes of δ. The lines are fixed - i.e. spatial mesh refinement does not occur on (t_k,t_{k+1}). They are discretized with variable step sizes $\{\delta t_i\}_{i=1,2,\ldots}$, which are sequentially chosen to control the local time discretization errors arising in the numerical integration of the ODE-IVP. Time integration and step-size selection are carried out by the variable-order backward differentiation formula algorithm implemented in the LSODI subroutine package of Hindmarsh and Painter (cf. Refs. 13,14). For our purposes, it suffices to consider LSODI in one of its simplest modes of operation: the implicit Euler method is used in conjunction with a modified Newton method to advance one time step δt_i, the local error is estimated with function values at t and t - δt_i, and δt_i and the criterion for stopping the Newton iteration depend on an input tolerance tol (chosen in the MOL to be much smaller than TOL in objective ii).

When time t_{k+1} is reached, the MOL core algorithm is completed with one to three operations. First, the space discretization error at t_{k+1} is estimated and a decision is made as to whether δ should be changed. If mesh modification occurs, some elements are uniformly refined and others are coalesced, with the resulting element distribution determined by the distribution of local space discretization error indicators at time t_{k+1} and by certain global information having dimension proportional to that of the coarse

grid Δ_x. The final operation is determination of new (initial) solution data if mesh modification has occurred.

In contrast to the MOL, the LRM algorithm for $(a,b)\times(t_k,t_{k+1})$ does not decompose space and time discretization into two distinct phases; it uses different time discretizations in different regions of space; and it is recursive in nature, with reintegration carried out over space-time regions of high error. The LRM core algorithm begins with spatial solution values at time t_k and a rectangular space-time net, defined by t_k, t_{k+1} and the coarse space grid Δ_x. Discretization of Eq. (19.1) initially is carried out using piecewise-bilinear space-time finite elements on this net. This results in a system of nonlinear algebraic equations which are solved via Newton's method. Using a space-time discretization error estimator to flag regions of high error, some of the elements in the original net are uniformly refined, and Eq. (19.1) is locally reintegrated on these refined elements. This process is recursively continued until error tolerances are satisfied on the finest grids. The core algorithm terminates by prescribing solution data at time t_{k+1}. An example of a sequence of grids that might be produced by the LRM core algorithm are pictured in Figure 1.

In the remainder of this chapter, we examine the MOL (Section 2) and the LRM (Section 3) more closely and further compare the two (Section 4). The comparisons presented here are qualitative. Quantitative comparison of the two methods could be very misleading. Different norms are used in the two; structural dissimilarities diminish the value of computational work models based on number of elements, time steps or function evaluations; and comparison of central processing times would reflect implementation styles and computing environments which we do not at this time wish to emphasize. We

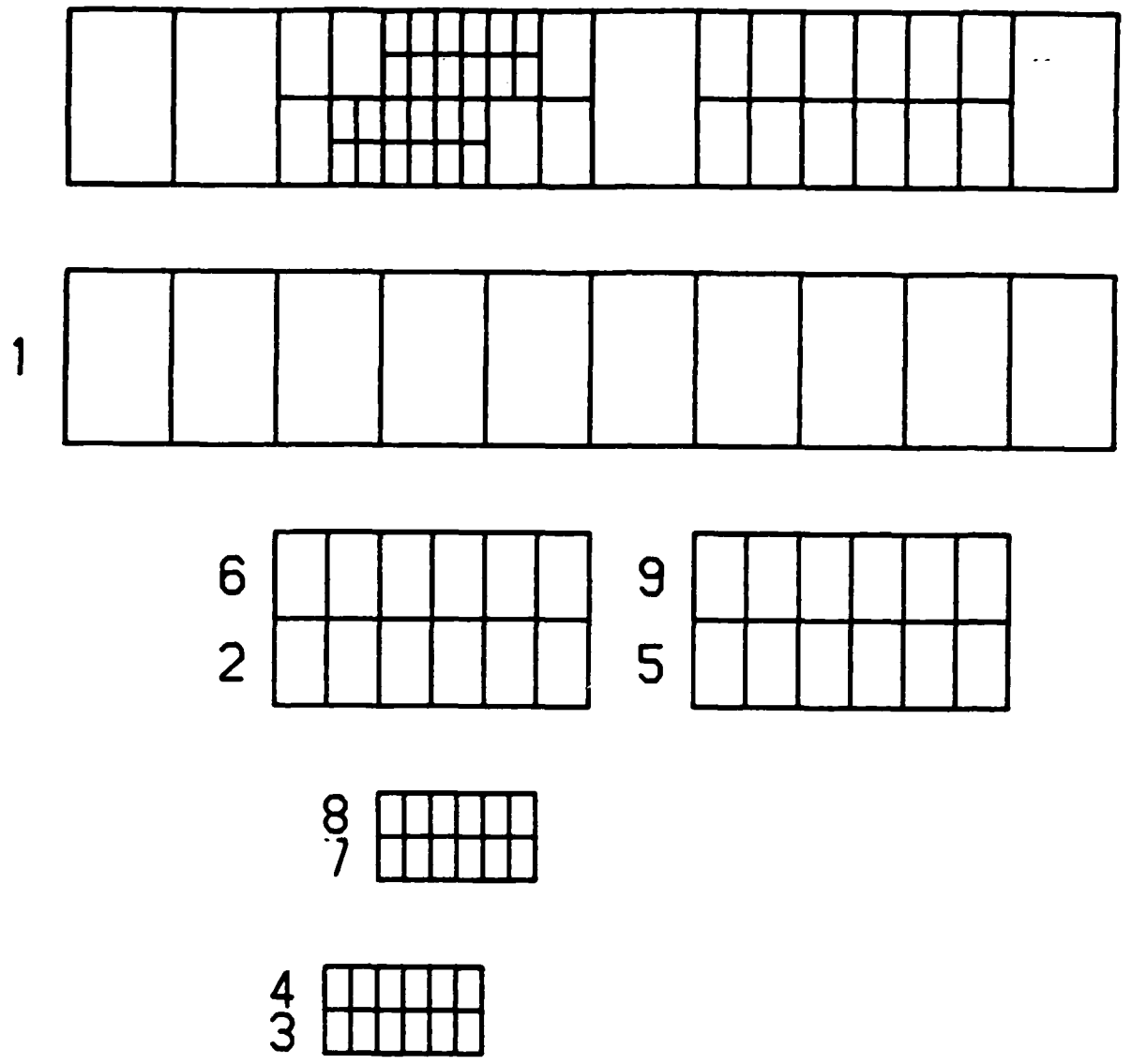


Figure 1. Typical set of local refinement grids for the solution of Eq. (19.1) on a coarse time step. The numbers indicate the order in which the solution is computed on the tree structure.

wish to call attention to issues rather than what might be viewed as intermediate solutions.

What are some of the important issues arising in these or any other adaptive methods of similar type for time-dependent PDEs? To a certain extent, the important issues all arise from the fact that a one-parameter family of problems must be solved, in contrast to the situation for adaptive methods for linear elliptic boundary value problems. Data structures must support refinement and derefinement, measures must be taken to prevent the propagation of errors, control strategies must take care not to introduce instabilities through assignment of initial and boundary data at mesh interfaces, and the role of prediction could be crucial to the success of a method.

In Section 3 we present a data structure for the LRM that is different from others described in this text. The data structure used in the MOL is quite simple and is not discussed. Readers interested in a data structure for two-dimensional problems of similar type which supports derefinement and utilizes nested dissection are referred to in Ref. 11. In Section 2, we look closely at one estimator for the MOL and some of the factors that affect its performance. The role of prediction is most germaine to the MOL, since integration in the MOL proceeds only in the positive time direction. We discuss how to incorporate pattern recognition ideas in the prediction process in order to control errors in a stable manner. Assignment of data at mesh interfaces is important in both methods, but most important in the LRM. For this reason, this topic is emphasized in Section 3.

2. THE METHOD OF LINES

2.1. Preliminaries

For simplicity, notation will be introduced only for a scalar model version of Eq. (19.1):

$$u_t + cu_x - (du_x)_x + f = 0 , \tag{19.4}$$

where $f = f(x,t,u)$, c and d are constant with $d > 0$, and where the solution u satisfies zero Dirichlet conditions at the spatial end points a and b. Let

$$\delta := \{a = x_0 < x_1 < \ldots < x_n = b\} \tag{19.5}$$

denote an arbitrary space grid with local grid sizes

$$h_j = x_j - x_{j-1} , \text{ for } j = 1, 2, \ldots, n . \tag{19.6}$$

and $S(\delta)$ be the finite element space of continuous functions which vanish at a and b and are linear on each (x_{j-1}, x_j).

The principal input for the MOL at time $t_0 = 0$ consists of

i. a space grid δ_0 whose nodes include those of the coarse grid Δ_x,
ii. course uniform space and time grids Δ_x and Δ_t (cf. Eqs. (19.2, 3)),
iii. a space discretization error tolerance TOL > 0, and
iv. a local time discretization error tolerance tol > 0.

The principal output from the method consists of

i. error estimates $\{E(t_k)\}_{k = 1, 2, \ldots, K}$,
ii. times$\{t_p^*\} \subset \{t_k\}$ at which space mesh modification occurs,
iii. corresponding space grids $\{\delta_p\}$, which each contain Δ_x, and

iv. an approximation U of u which is in $S(\delta_p)$ for $t \in [t_p^*, t_{p+1}^*)$, where $U(\cdot, t_p^*)$ is obtained by linear interpolation.

There typically are a great many ODE-IVP integration time steps taken on each interval (t_k, t_{k+1}), and many of these intervals in each (t_p^*, t_{p+1}^*). The ODE time step size sequence generated by LSODI generally increases on each (t_p^*, t_{p+1}^*), and it increases most rapidly just after being restarted at t_p^*, when LSODI is permitted to choose the initial step size. This step size sequence depends on the value of tol, which is chosen a priori to be small with respect to TOL (say 0.01TOL). The question of how small it must be in order for global time and space discretization errors to be comparable is very difficult to answer and is not addressed here. For our discussions, tol is assumed to be such that errors due to the space grids $\{\delta_p\}$ dominate those due to time discretization. The estimates $\{E(t_k)\}$ are space discretization error estimates which are assumed also to provide reasonable estimates of the total error.

There are two refinement decisions which are made for each space grid in the MOL: when to change the grid and how to construct a new grid. The strategy for selecting the times $\{t_p^*\}$ is quite simple. Time t_k is a "regridding" time if and only if $E(t_k)$ exceeds a present threshold, which is typically 0.9TOL. Mesh modification is carried out so that E(t), for t just beyond t_p^*, is lowered to a smaller value γTOL. The next regridding time will thus occur when E has grown by a factor of at least $0.9/\gamma$. While γ is normally adjusted during PDE integration, it will be considered as fixed (0.7 say) when it reenters the discussion. One should note that by allowing the error to grow and using a fixed stopping criterion (e.g., 0.9TOL), it is assumed that the PDEs being solved are sufficiently dissipative, and that the

error can be "recovered" in the future via mesh refinement.

2.2. Error Estimation

The accuracy measure of interest for the model problem (19.4) is

$$|||e(t)||| = [\int_a^b de_x^2(x,t)dx]^{\frac{1}{2}} . \tag{19.7}$$

Letting δ of the form (19.5,6) be the space grid in use at time t, the estimate E(t) of $|||e(t)|||$ is

$$E(t) = [\sum_{j=1}^{n} \eta_j^2(t)]^{\frac{1}{2}} , \tag{19.8}$$

where the indicators $\{\eta_j(t)\}$ are defined by

$$\eta_j^2(t) = (h_j^2/12d)\int_{x_{j-1}}^{x_j} [U_t(x,t) + cU_x(x,t) + f(x,t,U(x,t))]^2 dx . \tag{19.9}$$

The indicators provide information on the local spatial accuracy of the method and on the local spatial behavior of higher solution derivatives appearing in the leading terms of an error expansion. Recognizing the integrand above as the residual of the PDE (19.4) (neglecting discontinuities at the nodes of δ), we can extract this latter information by computing values of the function

$$w(x,t) := [12\eta_j^2/h_j^3]^{1/3} \; ; \; x \in (x_{j-1}, x_j) \tag{19.10}$$

$$\sim [du_{xx}^2(x,t)]^{1/3} .$$

The accuracy of this local information is generally worse than the averaged global spatial information given by E(t). The accuracy or performance of E can be assessed with the effectivity index

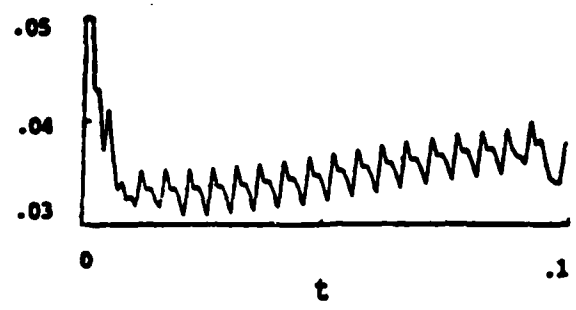
$$\theta(t) = E(t)/|||e(t)||| . \tag{19.11}$$

Theory and practice suggest that $\max_t |\theta(t) - 1| \to 0$ as $\max_t |||e(t)||| \to 0$ for a broad class of problems, with order and rate of convergence depending on many factors. Problem-based factors include the diffusivity of the system (i.e., the size of d), the relative strength of convection (i.e., the size of c/d), and the relative strength of reaction or source terms (i.e., the size of f_u/d). Mesh-based factors include the regularity of the space grids and the frequency with which they are changed. In any single PDE integration from t_0 to T, the accumulation in time of pollution effects related to any one of these factors can degrade the performance of E, nullifying the success of subsequent regriddings in the same run. And this can happen even if each ODE-IVP is solved exactly.

We illustrate how an inappropriate MOL refinement strategy can destroy the error control it attempts to achieve by solving the simple forced linear heat equation of Example 2 in Section 3. Two runs were made: one with 200 fixed uniform elements and the other where regridding was forced to occur every 0.001 time units. In the latter run, a local error indicator equilibration strategy was used to regrid and in both, computation of the load vector and time integration was very nearly exact.

With 200 fixed elements, the relative $|||\cdot|||$ - error and the effectivity index varied only slightly from the values 0.051 and 1.002, respectively, during the time the solution front was entirely within the space interval (0,1). Results taken from the latter run are shown in Figure 2. $|||e|||$ and θ oscillate in time, but more importantly, they become worse as time passes. θ - values in the range pictured in the lower graph of Figure 2 are certainly reasonable, but compare them with those from the less accurate (!) fixed element run.

$$\frac{|||e|||}{|||u|||}$$



θ

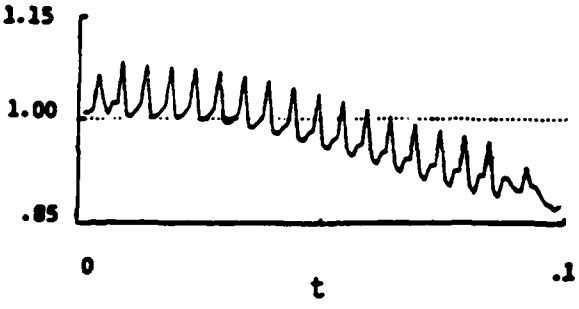


Figure 2. Relative error (top) and effectivity index, $\theta = E/|||e|||$, (bottom) for the heat conduction problem described in Example 2.

The onset of instability we see is the result of regridding too frequently with respect to the sizes and distributions of the spatial elements and would result with any reasonable assigned initial data when regridding occurs. By fixing 0.001 as the time between regriddings, requesting more accuracy (i.e., smaller h_j), and using the same equidistribution strategy in another experiment, the appearance of instability went away, as predicted by theory.

The risk of introducing undetected instabilities is taken by any method which employs purely local strategies or whose performance depends strongly on many input parameters. Such instabilities have been reported for moving finite element schemes in Refs. 10 and 17. In the MOL described in this chapter, this risk is reduced through the use of the course grids Δ_x and Δ_t, which would remain fixed if a problem were solved with many error tolerances $\{TOL_q, tol_q\}_{q=1,2,\ldots}$. The role of Δ_t was briefly explained at the end of Section 2.1. The role of Δ_x is best understood after examining the grid properties which change when mesh modification occurs.

2.3. Grid Shape and Intensity

Two properties of the grid δ are changed when it undergoes refinement and derefinement: shape and intensity. Introducing the grid function

$$\xi_\delta(x) := \begin{cases} 1/h_j \ , \ x \in (x_{j-1}, x_j) \ , \ j = 1, 2, \ldots, n \\ (1/h_j + 1/h_{j+1})/2 \ , \ x = x_j \ , \ j = 1, 2, \ldots, n-1 \end{cases} \tag{19.12}$$

we can define the shape of δ via the graph of the function

$$R_\delta(z_1, z_2) = \xi_\delta(z_1)/\xi_\delta(z_2) \ ; \ z_1, z_2 \in (a,b) \tag{19.13}$$

and the intensity of δ as

$$I_\delta := \int_a^b \xi_\delta(x)dx \ . \tag{19.14}$$

Notions related to shape and intensity are often used to describe grids for linear elliptic PDEs (cf. Ref. 3), but seldom are used in local refinement strategies to construct them, since both shape and intensity necessarily change. It is easily checked that δ has intensity $I_\delta = n$, and that grid intensity is unchanged if the same number of elements are both added and removed to change a grid's shape.

All sequential algorithms for time-dependent or parameterized nonlinear PDEs must select grid shapes which will work in the future - and the "future" for such algorithms is generally much farther ahead of the present than for which local extrapolations can be used. The example of the previous section showed how errors could grow undetected if one type of local extrapolation was used to predict appropriate grid shapes. What was not shown was the cost of such an algorithm. In practice, the expense of frequently interrupting the time integrator to construct new grids can completely overshadow a loss of accuracy.

The strategy used to construct a grid here is to attack the grid shape problem directly. A model grid function ξ is first explicitly constructed. It then is magnified (multiplied by an iteratively chosen constant) to yield a model grid intensity I defined as in (19.14). Lastly, refinement and derefinement are carried out so that the resulting grid has a shape closely resembling that of ξ (defined as in (19.12)), with the total number of elements approximately equal to I. More specific details of the actual construction can be found in Ref. 4.

Let us briefly see how the shape and intensity of a grid δ, constructed at time t^- and retained until time t^+, affect accuracy and cost on (t^-,t^+). To this end, we can use the functions w and ξ_δ (cf. (19.10), (19.12)) in the definition (19.8) of E to get

$$E^2(t) = (1/12)\int_a^b [w^3(x,t)/\xi_\delta^2(x)]\, dx \ , \ t \in (t^-,t^+) \ . \qquad (19.15)$$

Recalling from Section 2.1 that δ was constructed so that $E(t^-) = \gamma TOL$, it is noted that the least intensity (number of elements) required to do this would have been that of the grid whose shape coincides with the shape of $w(\bullet,t^-)$. This is the local error equidistributing grid, obtained by minimizing the right-hand side of (19.15) at $t = t^-$ in a manner similar to that described in Ref. 3. Ignoring the costs associated with frequent regridding, this grid might be described as the high risk, low cost alternative, since error control could be lost, but the ODE-IVP for $t \geq t^-$ would have the smallest possible size. On the other hand, a uniform grid which lowered $E(t^-)$ to the same value might be called the low risk, high cost alternative, since the maximum intensity of all reasonable grids would be required, but unpredictable changes in the shape of $w(\bullet,t)$ would be accounted for, and E would not suddenly exceed the threshold value 0.9TOL. The present method for selecting a model grid function ξ at t^+ tries to balance risk and cost in a general way by using a small amount of information collected at t^- and t^+ on the coarse space grid Δ_x.

2.4. Pattern Recognition and Grid Shape Prediction

In order to predict the future, all reasonable algorithms of the present type must employ heuristics, since accurate, inexpensively obtained local information just does not exist. These are usually justified using physical

reasoning or via global simulations of local mathematical expansions. If such prediction processes can ever be quantitatively assessed, compared, and justified in a general way, the first step may be to see them for what they are - pattern recognition processes. Such a process consists of three (generally nondistinct) stages:

Representation - reduction of (perhaps "noisy") data into a convenient and invariant form,

Feature Extraction - relevant measurements from the reduced data, and

Classification - decisions made by comparing feature values in an attempt to improve recognition or to avoid misrecognition.

Many adaptive strategies for nonlinear elliptic and time-dependent PDE's incorporate these stages in one form or another, including those described in 6 and 16.

The "pattern" we wish to predict (or recognize) here is the shape of the function

$$\bar{w}(x) := \max_{t \in [t^+, t^+ + \Delta T]} w(x,t) , \qquad (19.16)$$

where ΔT is unknown, but is comparable with $t^+ - t^-$. The attempt to do so consists of constructing a ξ which majorizes $w(\cdot, t^+)$ and has the shape of $\bar{w}$. Further reasoning for this choice can be found in Ref. 7.

In order to do this, the number N of "macro" elements in the course grid Δ_x is taken to be an integer multiple of 4, and Δ_x is treated as a 3-level grid, whose N/4 distinct level 1 macro elements each have size 4H and contain

2 level 2 macro elements having size 2H each and 4 level 3 macro elements having size H each. The size H (4H) is related to the maximum (minimum) risk of losing control over $|||e|||$ that one is a priori willing to take and the *minimum (maximum) price one is willing to pay to keep it.* The algorithm tries to manage risk and cost within this range on each level 1 "macro" element (X, X+4H) by first *reducing data* - The many pieces of data defining $w(\cdot,t^+)$ on (X, X+4H) are replaced by 3 piecewise constant functions $\{W_\mu(\cdot,t^+)\}_{\mu=1,2,3}$, where $W_\mu(\cdot,t^+)$ equals the maximum of $w(\cdot,t^+)$ on each of the level μ macro elements contained in (X, X+4H). In a similar manner, $\{W_\mu(\cdot,t^-)\}_{\mu=1,2,3}$ were computed and saved.

In solving many parabolic problems, it has been observed that there often is a correlation, on some scale, between spatial differences in $w(\cdot,t^-)$ and the way w subsequently grows in time. The algorithm checks if this was the case on $(X, X+4H)\times(t^-,t^+)$ by *extracting features* - Three measurements are taken:

$$M(\mu) = \int_X^{X+4H} |W_3(x,t^+) - W_\mu(x,t^-)|\, dx\ ,\ \mu = 1,\ 2,\ 3\ . \qquad (19.17)$$

The next step is to *classify* on (X, X+4H) - An active level μ^* is chosen as that for which $M(\mu^*)$ is the minimum of the three feature values. If $\mu^* = 3$, either w did not grow on $(X, X+4H)\times(t^-,t^+)$, or spatial differences at t^- were not the source of its growth. If $\mu^* = 2$ (1), then either a clear correlation exists on the scale 2H (4H), or w is evolving in a way which is unpredictable on any scale finer than this.

The final step is to decrease the dependence of the selection process on the geometry of the input macro grid Δ_X and to actually construct ξ. A macro subgrid consisting of the boundary nodes of every level 1 element and the

boundary nodes of every active level 2 or 3 element is formed (e.g., if $\mu^* = 2$ for some (X, X+4H), then X+2H is in the subgrid). ξ is then taken to be the piecewise-linear function on the subgrid whose j th nodal value is equal to the maximum of $w(\cdot,t^*)$ between the j - 1 st and j + 1 st subgrid nodes.

Figures 3 a and b illustrate how ξ was constructed in two simulations in which w was an exactly known unimodal function moving to the right. If such a w arose in an application of the MOL, the (implicit) prediction of movement would be carried over to the new spatial grid at t^* via $h(x) \sim 1/c\xi(x)$ for all $x \in [a,b]$ and some constant $c > 0$.

The pattern recognition ideas described above have been applied to several problems which demonstrate that the implicit shape prediction behaves in a stable manner. For some computational results see Ref. 7.

3. THE LOCAL REFINEMENT METHOD

The local refinement method is recursive and we begin by describing the discretization of Eq. (19.1) on an arbitrary strip $\alpha < x < \beta$, $p < t < q$. A finite element-Galerkin method is used with a uniform grid of n rectangular elements of size $(\beta - \alpha)/n$ by $(q - p)$. We refer to this grid as $R(\alpha,\beta,p,q,n,F,S)$, where F and S are pointers to the father and son grids discussed later.

We generate the discrete system on $R(\alpha,\beta,p,q,n,F,S)$ by approximating u by $U(x,t)$ and selecting test functions $V(x,t)$, where U and V are elements of a space of C^0 bilinear polynomials with respect to the grid R. We then take the inner product of Eq. (19.1) and V, replace u by U, and integrate any diffusive terms by parts to obtain

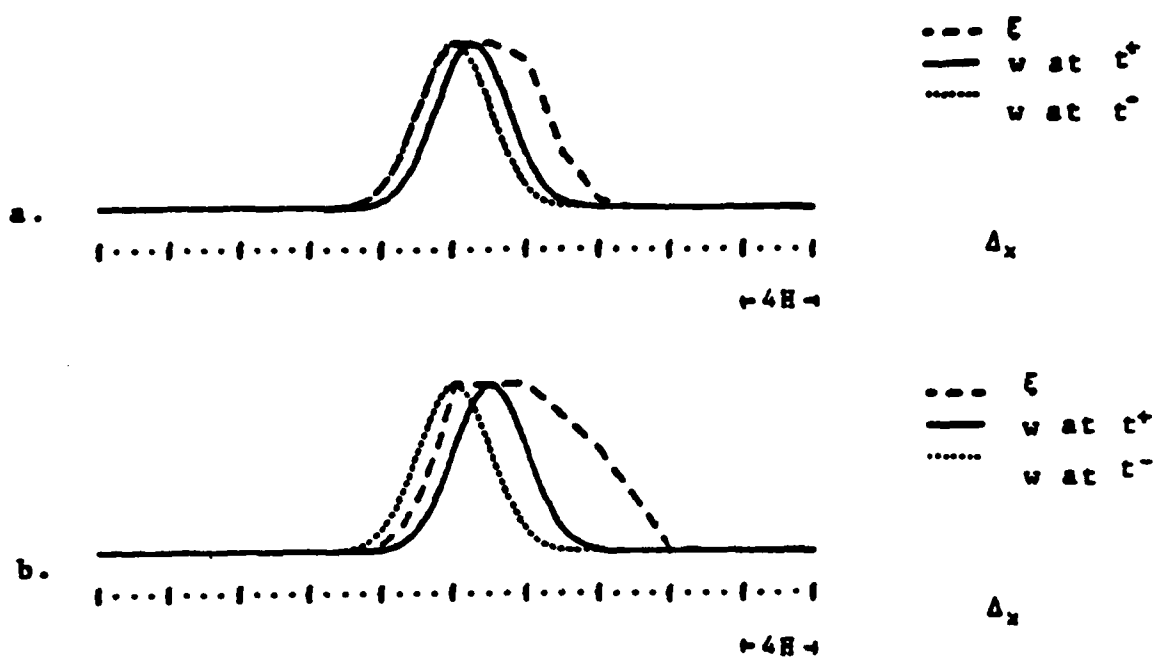


Figure 3. Prediction of model grid function ξ at time t^+.

$$\int_R [V^T U_t + V^T f(x,t,U,U_x) + V_x^T D(x,t,U)U_x]dxdt$$

$$-\int_p^q V^T D(x,t,U)U_x|_\alpha^\beta dt = 0 \ . \qquad (19.18)$$

Equation (19.18) must vanish for all bilinear functions V on the grid R. The integrals are approximated using a four-point Gauss quadrature rule and the resulting nonlinear system is solved by Newton iteration. Appropriate initial and boundary conditions for (19.18) are discussed later in this section.

We describe our local refinement procedure for solving problem (19.1, 18) for one time step (t_k, t_{k+1}) on a coarse grid with N elements, i.e, on $R(a,b,t_k,t_{k+1},N,0,S)$ (where the pointer F = 0 signifies that this grid has no father). To solve this problem we simply call the procedure "locref" with the arguments $R(a,b,t_k,t_{k+1},N,0,s)$, tol, tsub for each coarse grid time interval. A pseudo-PASCAL description of the procedure "locref" is shown in Figure 4.

```
procedure locref (R(α,β,p,q,n,F,S), tol, tsub)
   begin
      Solve the finite element equations (19.18) on R(α,β,p,q,n,F,S);
      Estimate the error on R(α,β,p,q,n,F,S);
      if error > tol then
         begin
            calculate where error > tol and return the son grids;
            for j := 1 to tsub do
               for i := 1 to number of sons do
                  begin
                     p[j] := p + (j-1)*(q-p)/tsub;
                     q[j] := p[j] + (q-p)/tsub;
                     locref (R(α[i],β[i],p[j],q[j],n[i],
                        R(α,β,p,q,n,F,S),S[i],tol,tsub)
                  end
         end
   end;
```

Figure 4. Algorithm for local refinement solution of (19.1, 18) on $R(\alpha,\beta,p,q,n,F,S)$ with an error tolerance of tol and dividing the local time step by tsub each time the error test is not satisfied.

The recursive algorithm locref sets up a tree structure of grids with $R(a,b,t_k,t_{k+1},N,0,S)$ being the root node and with the solution being generated by a preorder traversal of the tree at each local time step. For example, if the root grid is refined to give two subgrids and the time step is halved, then the problem is solved on the first subgrid on its first time step, then on the second subgrid on the same time step, then this procedure is repeated for the second time step (cf. Figure 1). The error is estimated by Richardson extrapolation, i.e., the problem is solved twice, once on a grid with half the space step and again on a grid with half the time step. The three solutions that are obtained at each original grid point are used to generate an error estimate. If this pointwise estimate exceeds the tolerance "tol", finer grids are added as leaf nodes to the tree. This procedure is similar to one used by Berger[4,6]. Note that tol includes both the temporal and spatial components of the discretization error. Richardson extrapolation is a simple procedure to implement since it uses the same finite element software as the solution. However, it is expensive and the approach used in the MOL seems simpler for finding the spatial component of the error. The temporal error could be estimated in the same manner as done with ODE software (e.g., LSODI); however, an appropriate relationship between temporal and spatial errors would have to be developed.

A problem common to both the MOL and LRM is the development of initial conditions when refining. In addition, the LRM approach needs internal boundary conditions on any grid where $\alpha \neq a$ or $\beta \neq b$. This is a difficult and crucial problem that is discussed for explicit finite difference methods by Berger[4,5]; however, it is largely unanswered for finite element applications. As we subsequently show in Example 1, instabilities or incorrect solutions can result if inappropriate conditions are specified.

In the LRM approach, initial conditions were obtained by saving the fine grid data at the end of each time step down to a given level λ in the tree. Initial conditions for finer grids were obtained by interpolation. Each grid in the first λ levels either has a linked list of the initial data directly associated with it or uses an initial data list of an ancestor grid. To find the value of the solution at some new initial point, the coordinate of that point is sequentially compared to values in the linked list until an interval containing the point is found so that interpolation can be used. We used either piecewise-linear interpolation or piecewise parabolic interpolation with shape preserving splines developed by McLaughlin[15]. We found some minor differences between linear and parabolic interpolation, but not enough to justify the additional effort of the shape preserving interpolants.

Storing data in a linked list and the necessary sequential search are costly and we are investigating more efficient procedures that use the natural ordering that already exists. Thus, since for each grid the values are sequentially stored we can simply store the location of the first value and the number of grid elements and use a binary search algorithm to find the necessary initial conditions.

At the present time, we prescribe internal Dirichlet boundary conditions by linearly interpolating from coarse to finer grids. A buffer zone of a given number of elements is added to each end of regions of high error that do not intersect the boundaries $x = a$ and b. If two buffer zones overlap or are separated from one another by one element, the two grids are joined. Similarly, if the buffer is only one element away from either a or b, that element is added to the grid.

We close this section with three examples using an experimental code implemented in FORTRAN-77 based on the LRM algorithm described above. All results were computed in double precision on an IBM 3081D computer.

Example 1. In order to illustrate the importance of adequately resolving initial conditions at each time step we solve the linear hyperbolic initial value problem

$$u_t + u_x = 0 \,, \tag{19.19}$$

$$u(x,0) = u^0(x) = \begin{cases} (1/2)(\cos(20\pi(x-0.45)) - 1) \,, & 0.35 < x < 0.75 \\ 0 \,, & \text{otherwise} \end{cases}$$

We solve this problem for one coarse time step of $\Delta t = 0.05$, 10 elements on $0 < x < 1$ and tol = 0.01. For small enough times the exact solution is $u^0(x-t)$. If initial conditions are interpolated from the coarse to the fine grid, the oscillations are missed and an incorrect solution is computed, possibly without a user realizing that there is anything wrong. However, upon saving initial values for the first 8 levels of the tree of grids the correct solution is calculated to the prescribed accuracy. The incorrect and correct solutions are shown at $t = 0.05$ in Figure 5.

Example 2. We consider the simple heat conduction problem

$$u_t = u_{xx} + f(x,t), \; t > 0, \; 0 \leq x \leq 1, \tag{19.20}$$

and select the source term $f(x,t)$ and the initial and Dirichlet boundary conditions so that the exact solution of (19.20) is

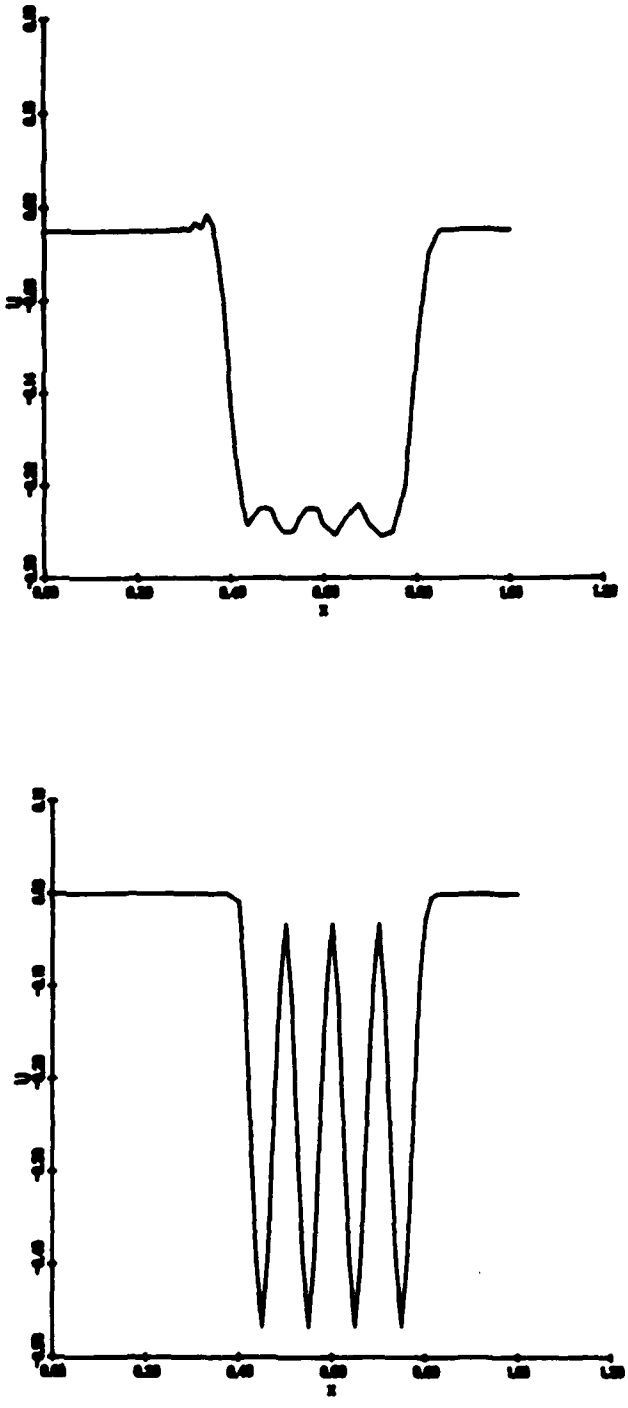


Figure 5. Solution of Example 1 at time t = 0.05 using interpolation from the coarse grid to the fine grid (top) and saving the initial values for the first 8 levels of the tree (bottom). The upper solution overlooks the oscillations and is incorrect.

$$u(x,t) = \tanh(40(x - 10t)) \qquad (19.21)$$

This solution represents a steep wave which travels from $x = 0$ to $x = 1$ with speed 10. For times greater than 0.1, the solution is essentially constant with value -1. We solved this problem using $t \leq 0.15$ with several error tolerances. For each tolerance, we present the number of elements used to reach time $t = 0.05$, the maximum pointwise error, and the error in strain energy (cf. Eq. (19.7)) in Table 1. The coarse grid consists of 20 uniform spatial elements and a time step of 0.05 for each tolerance.

tol	No. of Elements	$\|\|e\|\|_\infty$	$\|\|\|e\|\|\|$
0.1	5216	0.153	0.595
0.01	24296	0.0332	0.260
0.001	194672	0.0102	0.239

Table 1. Pointwise error tolerance, number of elements, pointwise error $\|\|e\|\|_\infty$, and error in strain energy $\|\|\|e\|\|\|$ at $t = 0.05$ for Example 2.

Table 1 shows that the number of elements to reach time 0.05 increases by factors of five and eight for each tenfold decrease in the tolerance. Similarly, the pointwise error decreases by factors of 4.6 and 3.3 for each tenfold decrease in the tolerance. Note, that we are controlling the local temporal error and reporting the global error, so we don't expect a correspondence between the tol and the pointwise error listed in Table 1. If uniform spatial and temporal refinement were used, we would expect the error to decrease by ten each time that the mesh was refined by ten. This is because the computation is dominated by the temporal error for the present

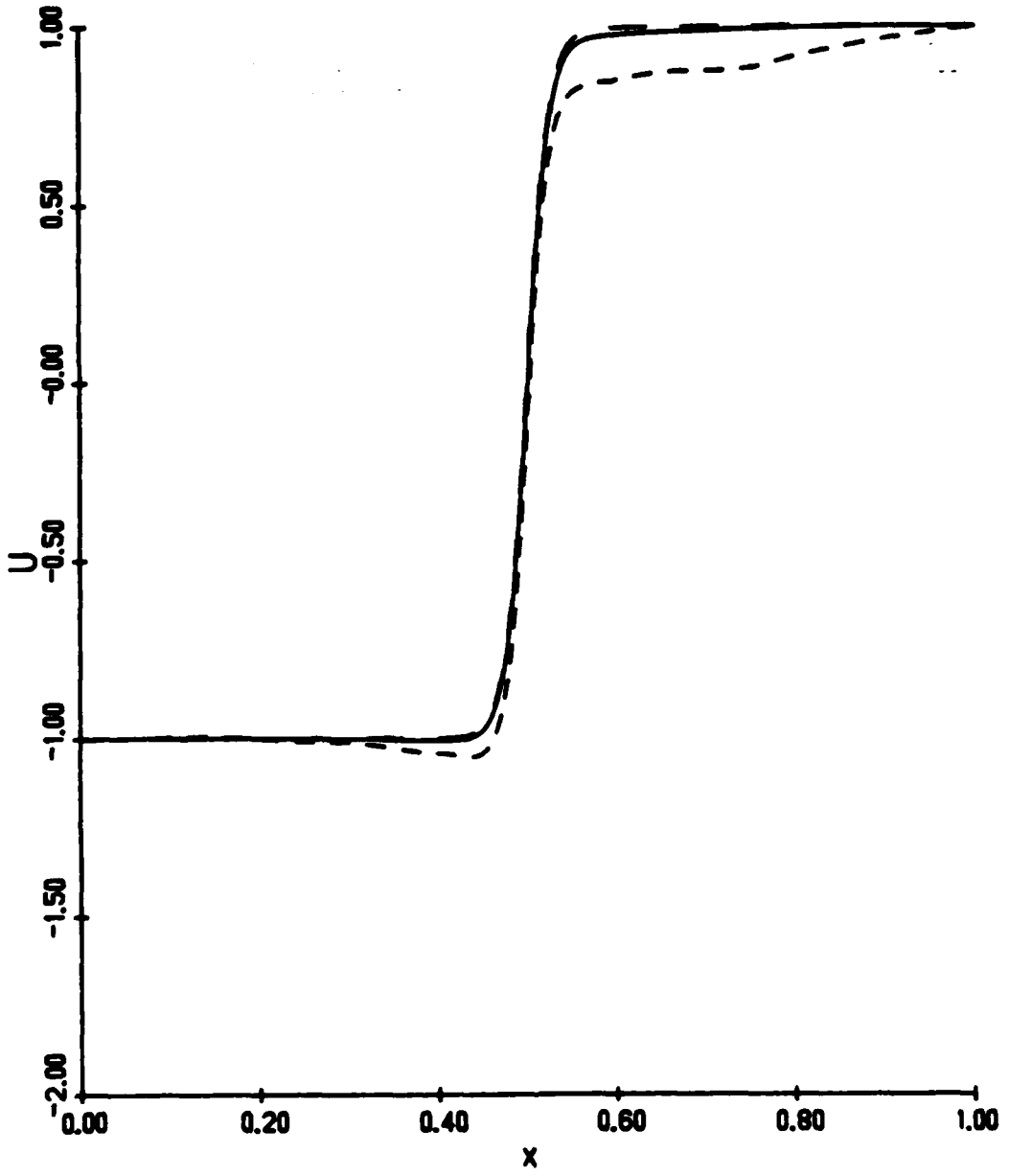


Figure 6. Solution of Example 2 at time t = 0.05 with tolerances of 0.1 (short dashes), 0.01 (long dashes), and 0.001 (solid).

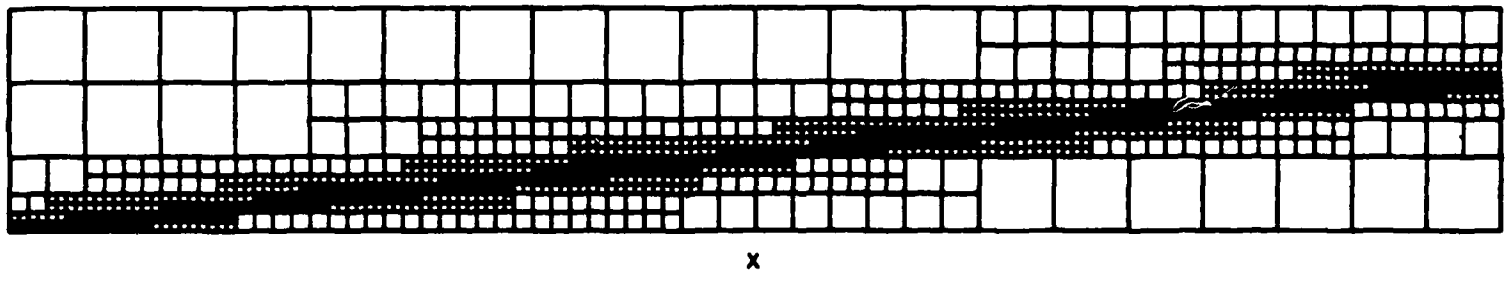


Figure 7. The grids generated by Example 2 on $0 < x < 1$ and $0 < t < 0.15$. The initial coarse mesh was 20 by 16 with $\Delta t = 0.05$.

problem. This would necessitate a one-hundred fold increase in the number of elements. Of course, uniform refinement by the same factors in space and time would not be judicious for the present methods and problem. Figure 6 shows the solution at t = 0.05 for the three tolerances, 0.1, 0.01, 0.001. There are spurious oscillations in the solution when tol = 0.1; however, these disappear as tol is decreased. Finally, Figure 7 shows the grids used for tol = 0.01 and $0 < t < 0.15$. We see that fine grids follow the wave across the interval and are only added to regions containing the wave front.

Example 3. We solve the model combustion problem

$$u_t + u_x - 2e^u = u_{xx} \ , \ 0 < x < 1 \ , \ 0 < t < 1 \ ,$$

$$u(x,0) = 0 \ , \ u(0,t)=0 \ , \ u_x(1,t) = 0 \ . \tag{19.22}$$

The exponential nonlinearity is typical in combustion problems having Arrhenius chemical kinetics. However, in this case the solution develops a "hot spot" at x = 1 and becomes infinite when t is approximately 0.85. We solve this problem for $t \leq 0.8$ for several tolerances and present the number of elements used as a function of the tolerance in Table 2. We begin each computation with a coarse grid of 20 elements and a time step of 0.05. The sequence of grids that were used to solve the problem with a tolerance of 0.001 are shown in Figure 8.

We see that the mesh is initially concentrated in the region near x = 0 where the curvature of the solution is largest. As time progresses and the curvature diminishes, excessive refinement is not necessary. Finally, as the solution begins to "blow-up" our algorithm generates a fine mesh only in the region near x = 1.

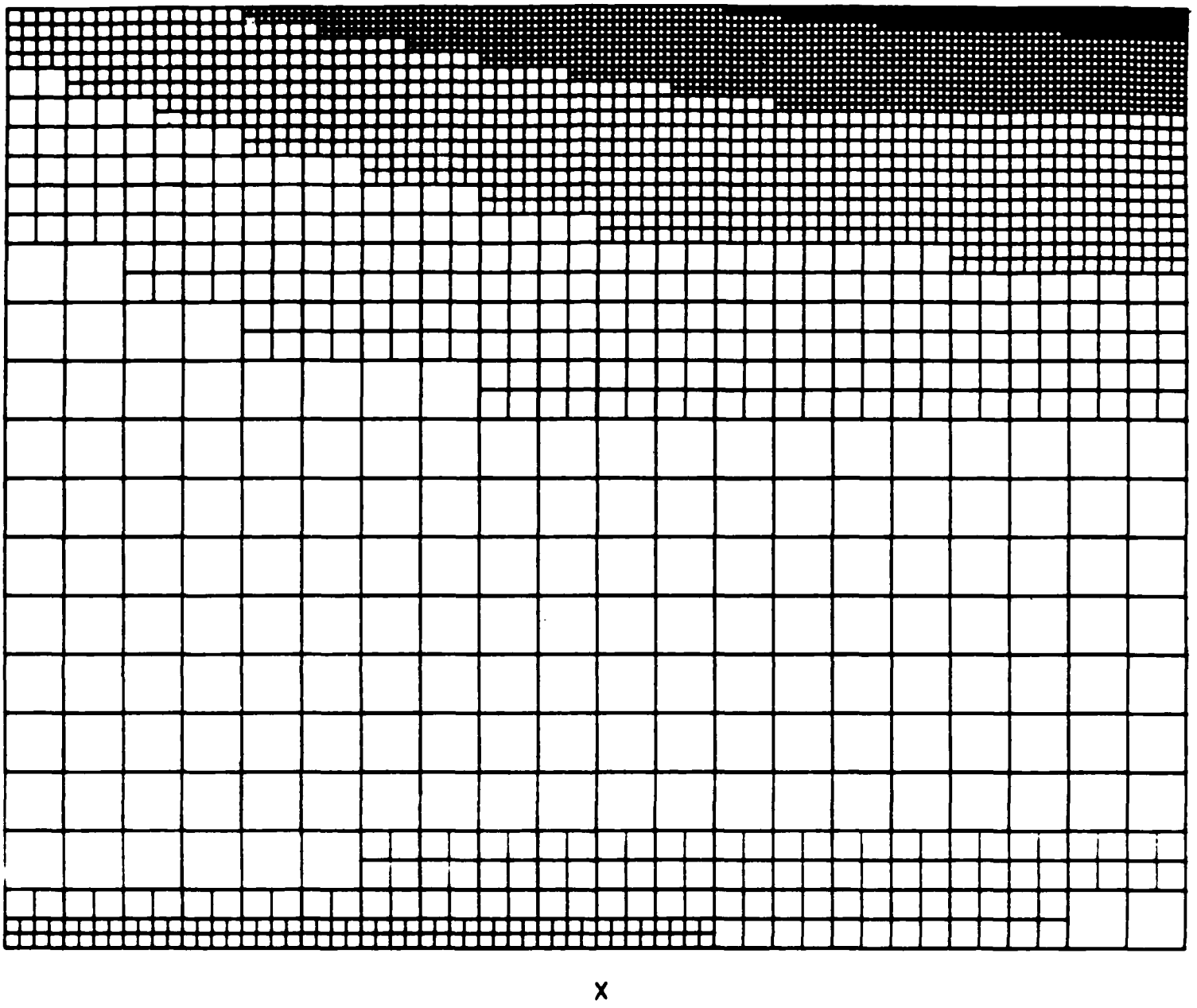


Figure 8. The grids generated by Example 3 on $0 < x < 1$ and $0 < t < 0.8$. The initial coarse mesh was 20 by 16 with $\Delta t = 0.05$.

tol	No. of Elements
0.1	68
0.01	296
0.001	2024

Table 2. Pointwise error tolerance and number of elements at t = 0.8 Example 3.

4. DISCUSSION

It was our original intent to provide some comparisons of the MOL and LRM codes; however, comparing computer codes is a difficult task and great care is necessary to define appropriate performance measures. Our codes are still under development and are not ready for the extensive comparison that has been performed for ordinary differential equations codes. Comparing computer times is always difficult in a multi-user environment. It is further affected by differences in implementation style and technique. Other performance measures that seem appropriate are comparing storage requirements, the number of finite elements or grids per time step, the effectiveness of the error estimators, and the theoretical and observed convergence rates for each code.

Quite generally, we observe some features, similarities and differences of the two methods. The MOL uses data structures that are simpler to implement in FORTRAN. The tree structures and the natural recursive nature of the LRM are more suited to languages such as PASCAL and LISP, yet these languages lack several important numerical capabilities and, at least

LISP, lacks portability. Different error control mechanisms are used in the two methods. The ODE integrator in the MOL is provided an error tolerance related to the spatial error tolerance, but the integrator does not utilize the fact that the ODEs arise in discretizing a PDE system. This, however, may not be optimal for some problems which are convection dominated or where highly local phenomena evolve, and the LRM may be more suitable. The pattern recognition techniques used in the MOL reduce the need to "back up" and reintegrate a time step. While there is no essential reason why the MOL technique could not include the ability to reject time steps, including pattern recognition in the LRM would destroy the storage and CPU advantages of using uniform grids. Perhaps, a compromise for LRM would be to use pattern recognition to determine a best uniform coarse grid for the subsequent time step. However, since the LRM rejects time steps and has the ability to back up, it is more difficult to deceive and less sensitive to coarse grid selection.

Since the MOL solves globally in space at every time step, it is never concerned with internal boundary conditions. As noted earlier this is an area of further work for the LRM. The error estimator used in the MOL procedure depends on the problems having diffusion. One point in favor of Richardson extrapolation is that it doesn't depend on problem type and this may allow the LRM to solve hyperbolic or mixed problems. Performance measures are much easier to specify for the MOL since the temporal error is assumed to be negligible. This is not true for the LRM and further study is needed to define quantities equivalent to those used by MOL, e.g., the effectivity index.

Neither the LRM nor MOL allow for mesh movement and, thus, may not be optimal for problems where dynamics are important, e.g., flame

propagation. A MOL approach that included both refinement and mesh movement was studied by Adjerid and Flaherty[1].

In this chapter, two adaptive methods with some similarities and differences were investigated. It is our intention to further examine the relationships between these methods and ultimately develop software that exploits the best features of each.

Acknowledgement. The second and third authors were partially supported by the U. S. Air Force Office of Scientific Research, Air Force Systems Command, USAF, under Grant Number AFOSR 80-0192 and the U. S. Army Research Office under Contract Number DAAG29-82-K-0197.

5. REFERENCES

1. S. Adjerid and J. E. Flaherty, "A moving finite element method for time dependent partial differential equations with error estimation and refinement," preprint, 1984.

2. I. Babuska, J. Chandra, and J. E. Flaherty (Eds.), *Adaptive Computational Methods for Partial Differential Equations*, SIAM, Philadelphia, 1983.

3. I. Babuska and W. C. Rheinboldt, "Analysis of optimal finite element meshes in R1," *Math. Comput.*, pp. 435-463, 1979.

4. M. J. Berger, "Adaptive mesh refinement for hyperbolic partial differential equations," Report No. STAN-CS-82-924, Department of Computer Science, Stanford University, 1982.

5. M. J. Berger, "Stability of interfaces with mesh refinement," Report

No. 83-42, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, 1983.

6. M. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *J. Comput. Phys.* 53 , p.484, 1984.

7. M. Bieterman and I. Babuska, "An adaptive method of lines with error control for parabolic equations of the reaction-diffusion type," U. Maryland IPST Tech. Report BN-1023, 1984.

8. M. Bieterman and I. Babuska, "The finite elememt method for parabolic equations, I. a posteriori error estimation," *Numer. Math.*, 40, pp. 339-371, 1982.

9. M. Bieterman and I. Babuska, "The finite element method for parabolic *equations, II. a posteriori error estimation and adaptive approach,*" *Numer. Math.*, 40, pp. 373-406, 1982.

10. J. M. Coyle, J. E. Flaherty and R. Ludwig, "On the stability of mesh equidistribution strategies for time dependent partial differential equations," submitted to *J. Comput. Physics*, 1984.

11. R. E. Ewing, "Adaptive mesh refinement in petroleum reservoir simulation," Chapter 16 of this text.

12. J. E. Flaherty and P. K. Moore, "A local refinement finite element method for differential equations," to appear in *Trans. Second Army Conf. of Applied Maths and Comput.*, 1984.

13. A. C. Hindmarsh, "Toward a systemized collection of ODE solvers," Report UCRL-874645, LLNL, Livermore, California, 1983.

14. A. C. Hindmarsh, "LSODE and LSODI, two new initial value ordinary differential equation solvers," *ACM Newsletter* 15, 1980.

15. H. W. McLaughlin, "Shape preserving planar interpolation: an algorithm," *IEEE Computer Graphics and Applics.* 3, pp. 58-67, 1983.

16. W. C. Rheinboldt, "Error estimation and adaptive techniques for nonlinear parameterized equations," Chapter 9 of this text.

17. R. F. Sincovec and S. Aslam, "Moving finite elements in 1-D," presented at the 1984 SIAM Summer Meeting, Seattle, Washington.

END

FILMED

11-85

DTIC